COBGDB - The GnuCOBOL TUI DEBUGGER / ANIMATOR

HOW TO USE COBGDB

Table of Contents

1. Introduction

COBGDB is a TUI (Text User Interface) application, programmed in C, designed to assist in animate and debugging GnuCOBOL TUI code **using GDB**.
The project is hosted at ***https://github.com/marcsosduma/cobgdb***

Very important: you don't need to know GDB and all its many commands (https://www.sourceware.org/gdb/). COBGDB has its own interface that is very simple to use and is responsible for interfacing the underlying GDB which is the real debug and animate engine but operates practically in a transparent and invisible way for the user.

The COBGDB application is based on the extension for Visual Studio Code (VSCode) created by Oleg Kunitsyn, which can be found on GitHub: *https://github.com/OlegKunitsyn/gnucobol-debug.*
warning: COBGDB is still currently under development.

At https://github.com/marcsosduma/cobgdb in the Windows subdirectory, the executable program cobgdb.exe for this operating system is available and ready to use.

To compile COBGDB from C source code on Windows, you can use MinGW.
The Makefile is configured to generate the program **`cobgdb.exe`** for both Windows and Linux.

1.1. Installing the debugger on Windows

On Windows, just download **`cobgdb.exe`** from following folder:
https://github.com/marcsosduma/cobgdb/tree/main/windows .

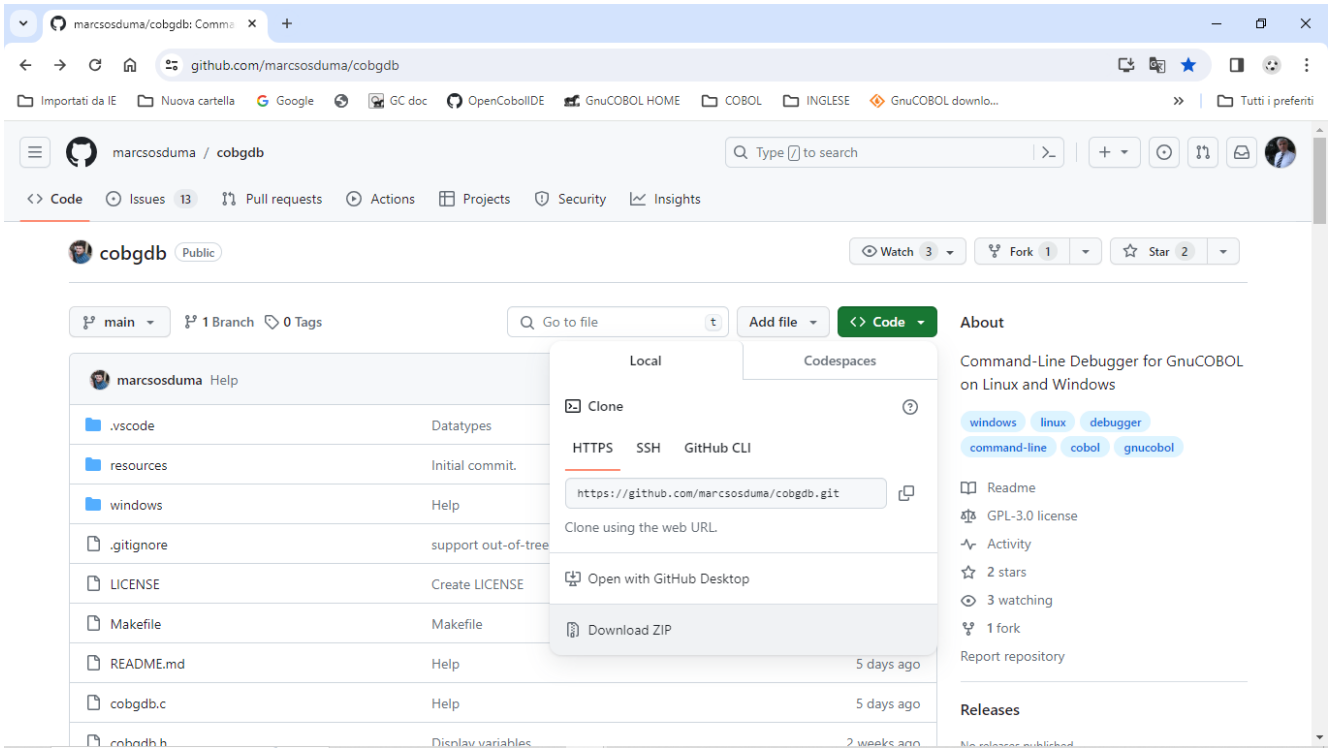
As an example you can put **`cobgdb.exe`** into the "**`bin`**" folder of your GnuCOBOL installation (the same folder where the GnuCOBOL compiler **`cobc.exe`** is located)

or

first install MinGW (Minimalist GNU for Windows).
Then execute the make ('mingw32-make' for Windows) command to compile the code from C source.

Note: if you have security problems downloading an .exe file then you can try download the entire repository with the following github button <> Code v --> *Download ZIP*

Then unzip the file and copy cobgdb.exe to the "bin" folder of your GnuCOBOL installation (the same folder where the GnuCOBOL compiler cobc.exe is located)



1.2. Compile and Debug GnuCOBOL programs

Compile and run a debugging session of the sample program using the following command:

cobgdb customer.cob -x -lpdcurses

Source code of customer.cob used also for following tutorial is at:
https://github.com/marcsosduma/cobgdb/tree/main/windows

Note: '**`-lpdcurses`**' is an instance of an argument that can be indirectly passed to '**`cobc`**' by '**`cobgdb`**,' even if it is not used by '**`cobgdb`**' itself.

or, other example for **`cobc`** parameters , use : **`cobgdb customer.cob -x -Tcustomer.txt`** .(-T creates a compilation list output into **`customer.txt`** file)

COBGDB takes one or more programs with COB or CBL extension as parameters and runs the GnuCOBOL compiler with the following format:

```
cobc -g -fsource-location -ftraceall -v -free -O0 -x prog.cob prog2.cob ...
```

To debug multiple programs, use COBGDB with the following syntax:

cobgdb prog.cob subprog1.cob subprog2.cob . . .

This will create a single prog.exe executable.

You can run GDB/GDBSERVER remotely using the "A" (Attach) key.
COBGDB will prompt you to provide the server and port in the format server: port or the PID of the application.
Example:

- localhost: 5555
- 9112

1.3. Main Commands

Cmd		Description	
?	Help	Show the HELP window and text	
B	Breakpoint	Toggle (Set or Unset) a breakpoint at a specific line of the Procedure Division code.	
R	Run	Runs the program from the first Cobol statement until a breakpoint is encountered. Always use this command to start a debugging session.	
C	Cursor	Runs the program until it reaches the selected line.	
N	Next	Runs the program until the next line but does not enter a subroutine executed by CALL or PERFORM.	
S	Step	Runs the program until the next line. If needed it enter a subroutine executed by CALL or PERFORM.	
G	Go	Continues the program execution until it encounters a stopping point: a breakpoint, end of the program, or the return from a subroutine (PERFORM / CALL).	
J	Jump	Ask for a line number and Runs the program until it reaches that line.	
V	Variables	Displays a new window with the set of variables for the running program. From this window you can also change the value of variables	
H	sHow	Shows a window with the values of variables on the selected line.	
F	File	When cobgdb is executed with more than one program, allows selecting one of those source file for debugging commands.	
A	Attach	Attach to GDBSERVER or Application PID.	
Q	Quit	Quits (Ends) the debugging session and the program.	

2. Tutorial - Sample Debugging Session

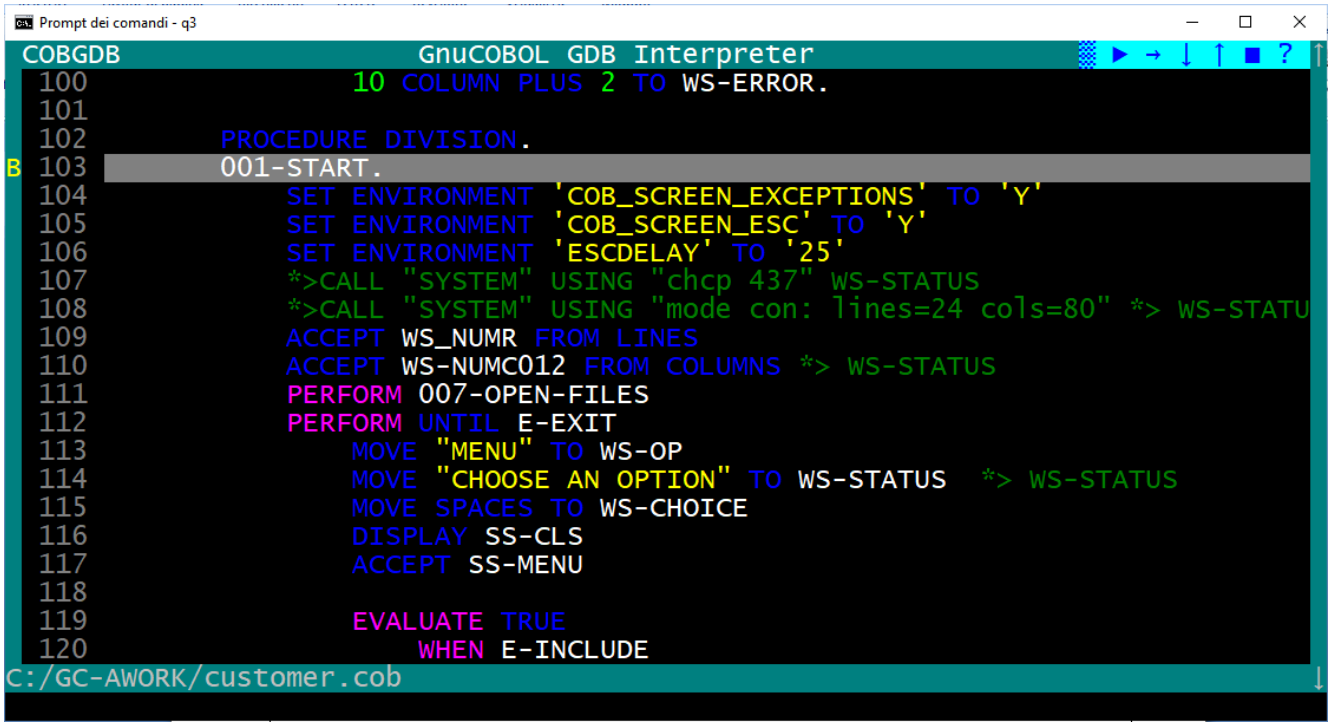
Following tutorial is on a Windows 10 platform using follwing version of GnuCOBOL:

```
cobc (GnuCOBOL) 3.2.0
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
Written by Keisuke Nishida, Roger While, Ron Norman, Simon Sobisch, Edward Hart
Built Jul 28 2023 16:07:38
Packaged Jul 28 2023 16:58:47 UTC
C version (MinGW) "13.1.0"
```

Downloaded from *https://www.arnoldtrembley.com/GnuCOBOL.htm*

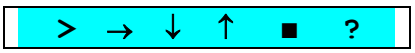
====== Version 3.2 ======
GnuCOBOL 3.2 (28Jul2023) MSYS2 64-bit GC32M-BDB-x64.7z -- MSYS2 64-bit GnuCOBOL 3.2 Final release **with full debugging support**. (95.4 Megabytes).
GnuCOBOL 3.2 (28Jul2023) MSYS2 32-bit GC32M-BDB-x32.7z -- MSYS2 32-bit GnuCOBOL 3.2 Final release **with full debugging support**. (96.1 Megabytes).

After executing **cobgdb customer.cob -x -lpdcurses** the application automatically insert a Breakpoint at first executable program statement of PROCEDURE DIVISON (see the B symbol at left of line 103 in this sample) and displays following screen:



You can scroll the source code window with cursor keys UP and DOWN, PG UP and PG DOWN or with mouse wheel or with mouse left click on the right scroll bar. Use cursor RIGHT and cursor LEFT to scroll horizontally,

In the upper right window corner there is a "button bar" where you can find some buttons (symbols):

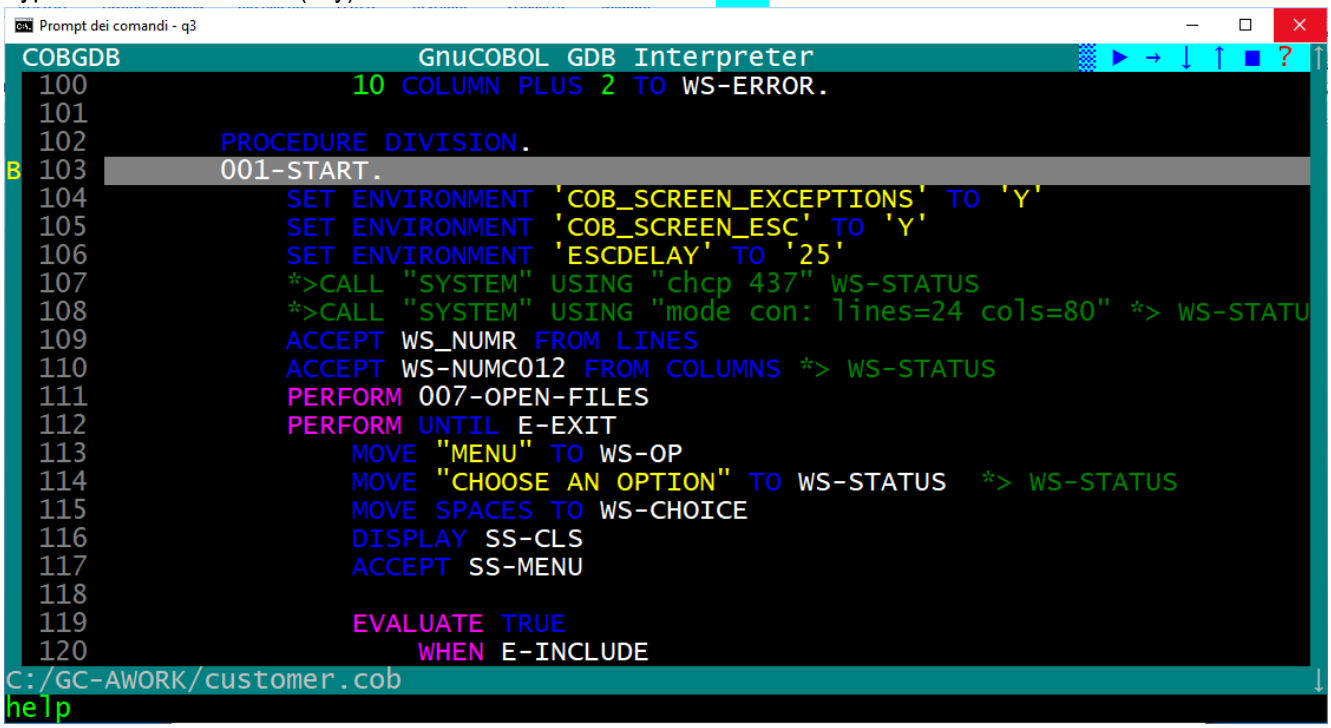


>	=	**Run** command
→	=	**Next** command
↓	=	**Step** command
↑	=	**Go** command
■	=	**Quit** command
?	=	**Help** command

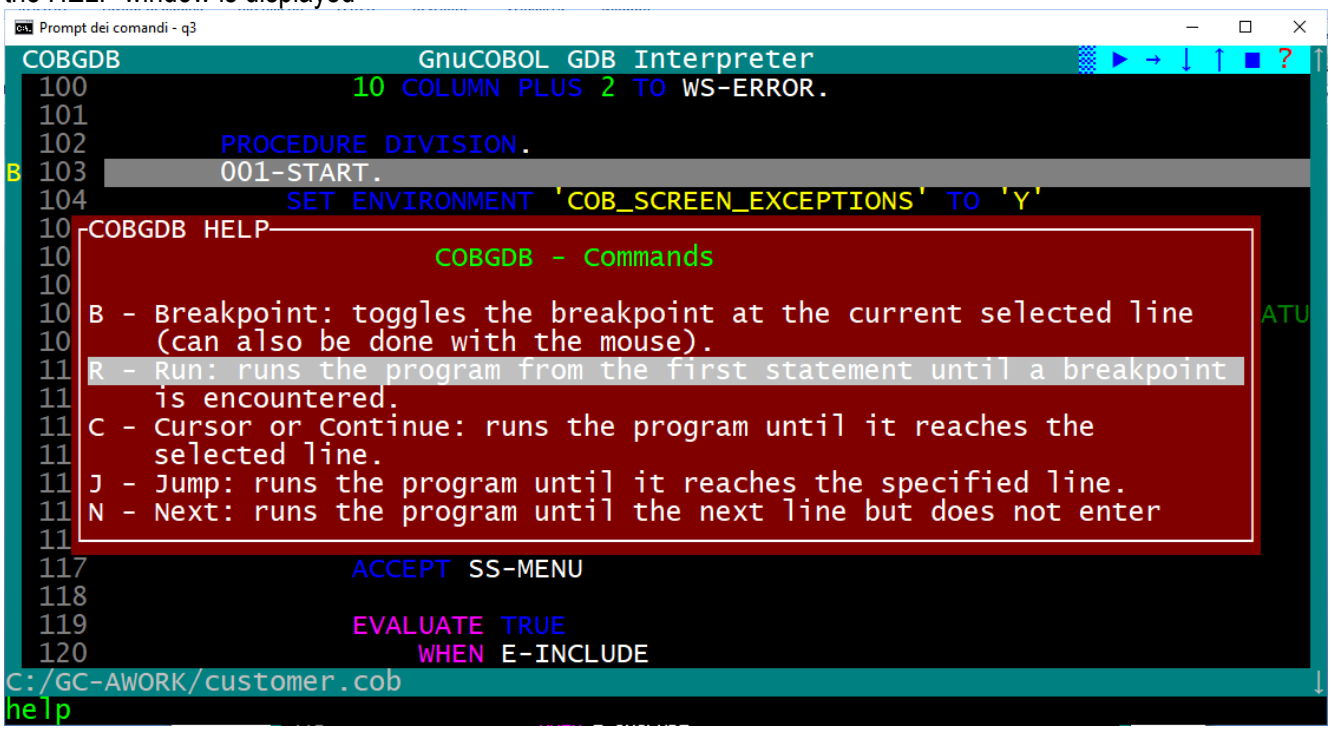
when you hover over one of these commands, you get the corresponding command description (like a tooltip) displayed at the bottom left of the screen.

2.1. Help Command

Type **?** HELP command (key) or left click with mouse on the **?** button:



the HELP window is displayed

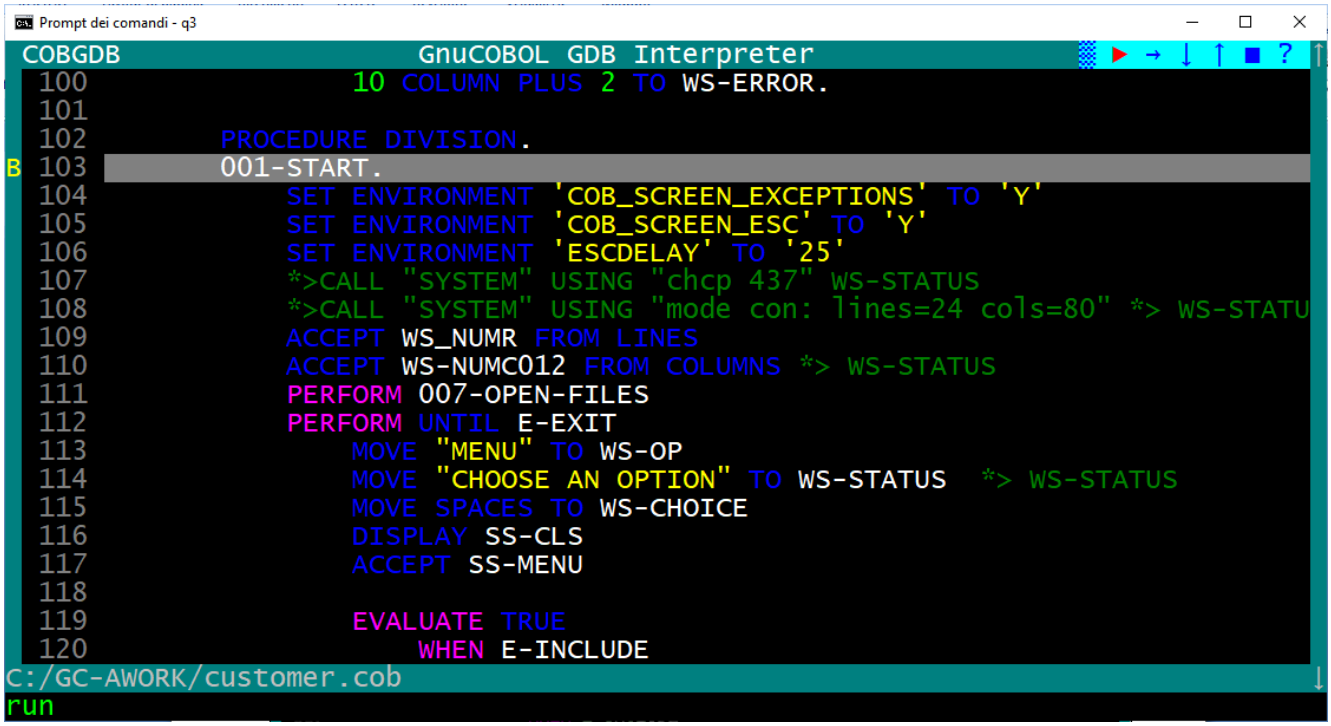


scroll the Help window with cursor keys UP and DOWN or mouse wheel.

Use ESC or Enter or left click to exit from this HELP window and return to debugging session.

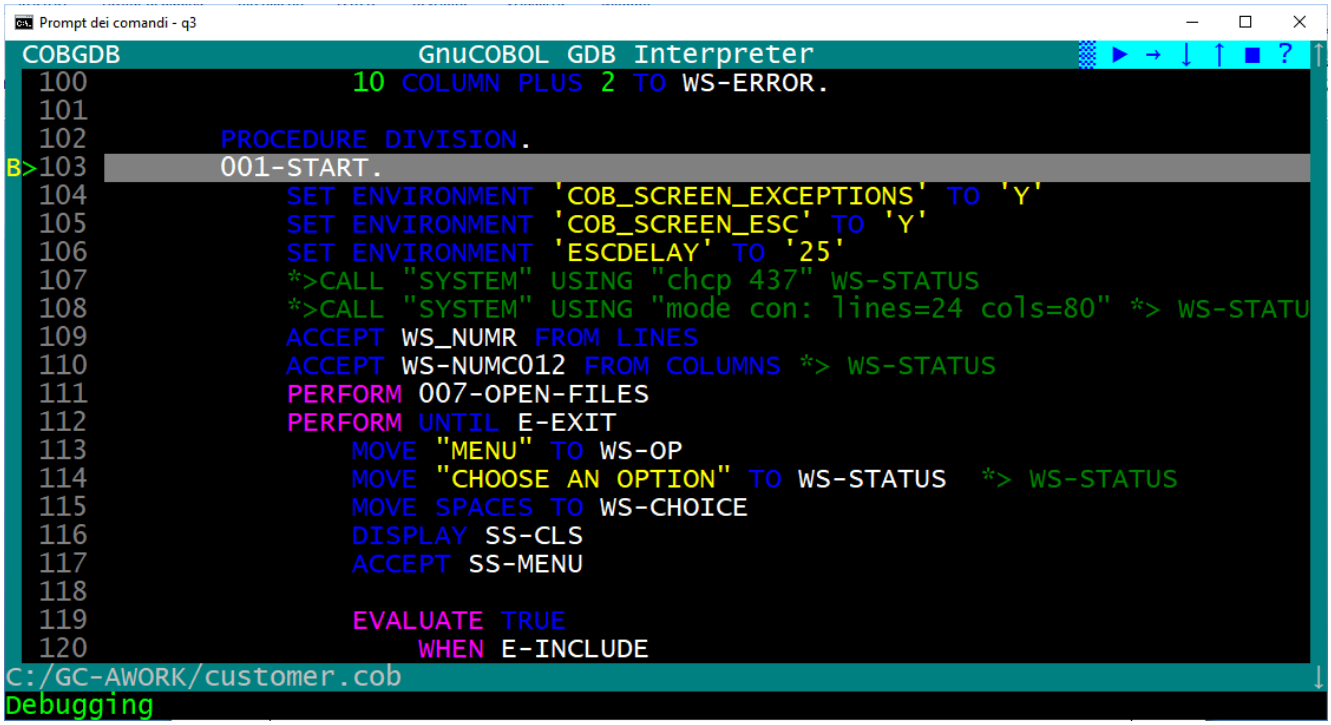
2.2. Run Command

To start executing the program and the debugging session from first program statement you always must use the "**R**" command (key) or left click with mouse on the **>** **Run** button



cobgdb opens the program terminal window (the application will run in this separate window.).

go back to the COBGDB screen, and you see a **>** green symbol on the left of statement where initial B Breakpoint is present (in our example is at line 103 **B> 103**):

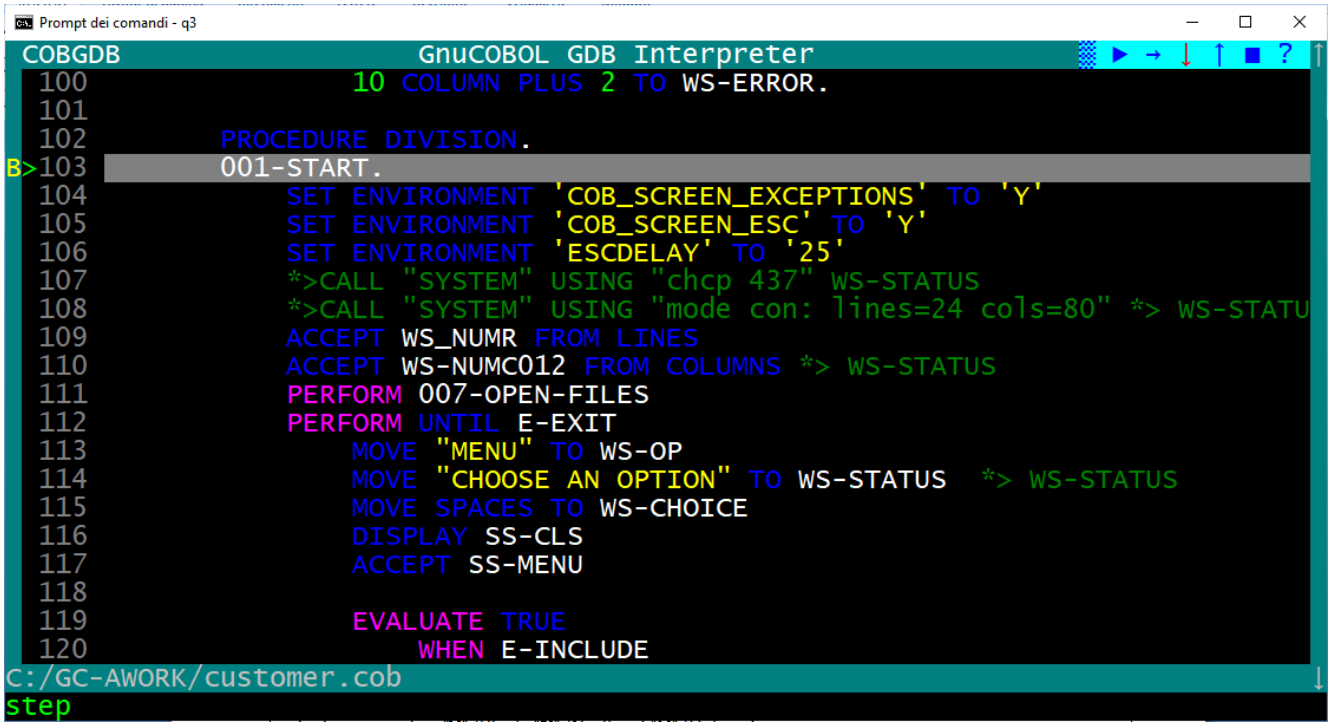


From that moment on, you can use all the commands (keys) or corresponding buttons to "animate" and debug the application, example: "**S**" (Step), "**N**" (Next) and "**G**" (Go) and so on.

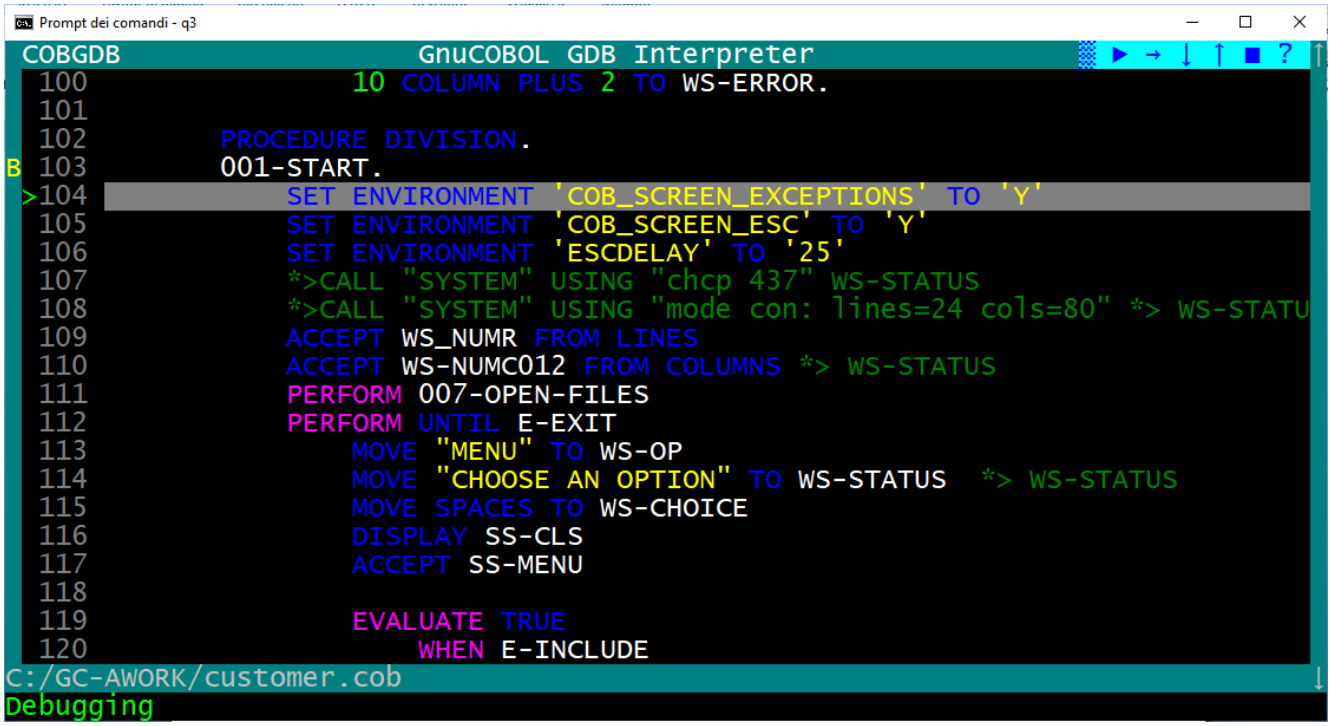
During the source code animation, the debugger automatically shows some pop-up windows with variables content from the line in execution.

2.3. Step Command

Proceed with **S** (Step) command or left click with mouse the ↓ button:.



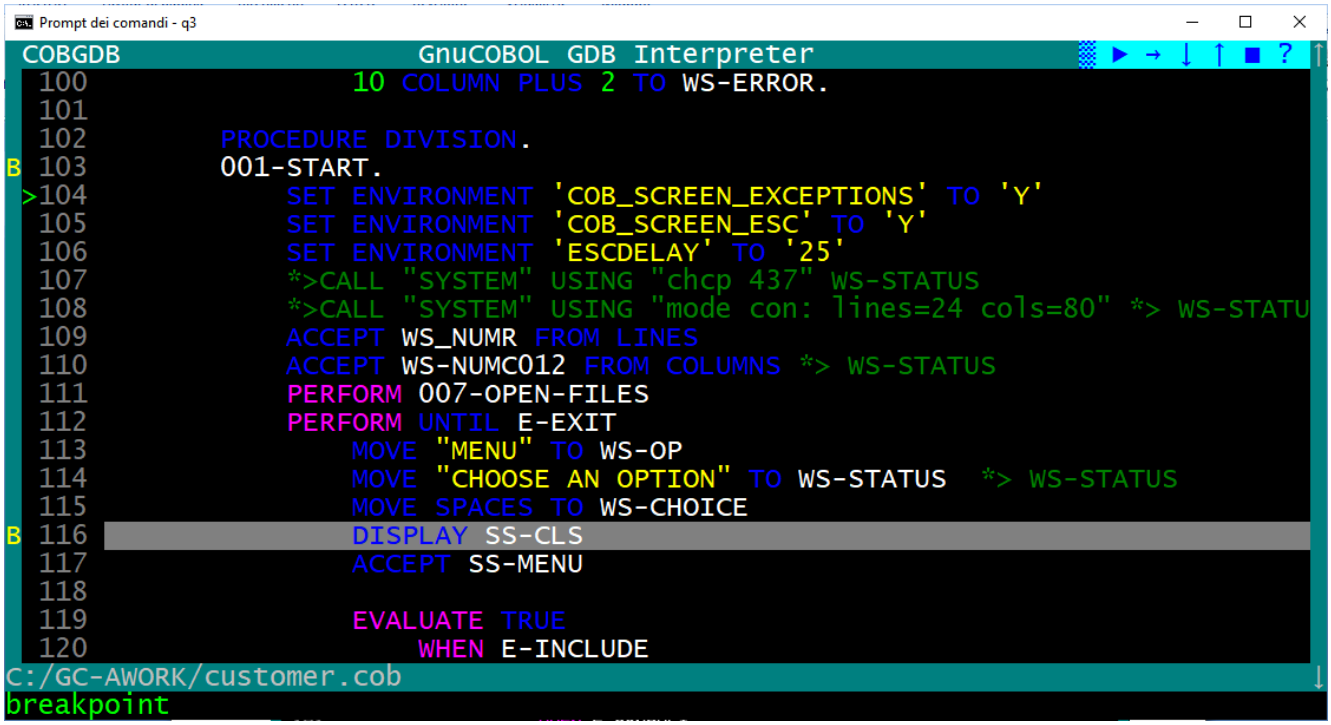
the **>** green symbol now is on the following line 104:



Now you can proceed with S command or N command or as an example:

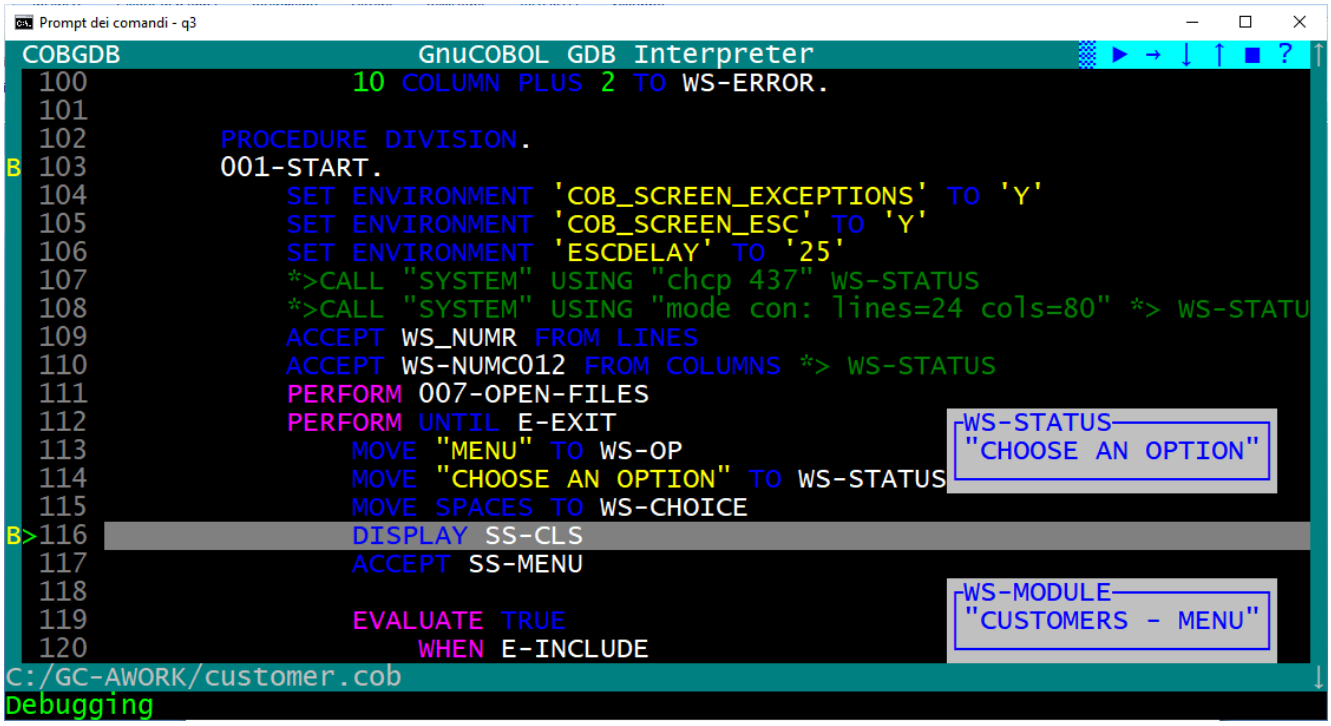
- Scroll with cursor down to select line 116 of Procedure Division and type "B" (to set a Breakpoint), (you also can simply click with mouse left button on the 116 row number)

- The application displays a "**B**" on the left of the line (type B again - or re-click - when you want to delete the Breakpoint, not do that at this moment)



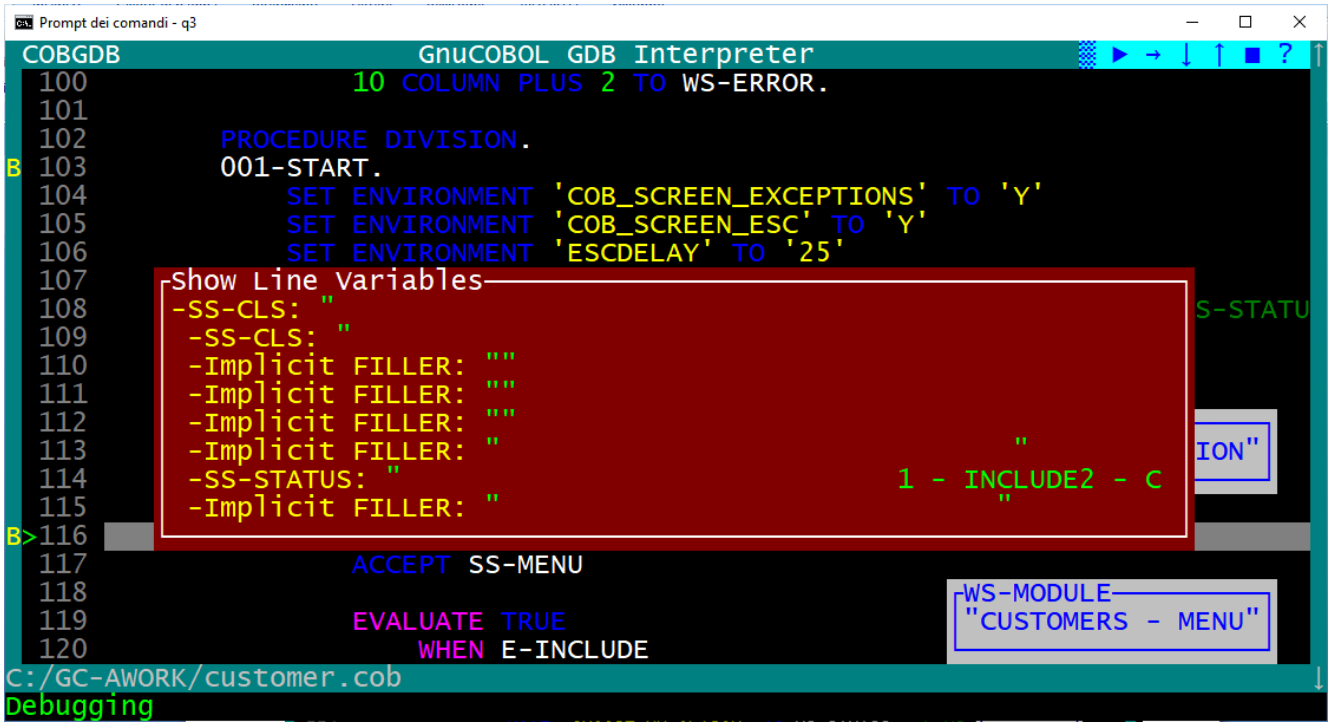
2.4. Go Command

Type **G** (Go) or left click with mouse the ↑ button to execute the program until a B Breakpoint is detected: the system reach the second breakpoint at line 116 and displays a green > symbol to the left of the line to be executed, see following screen:

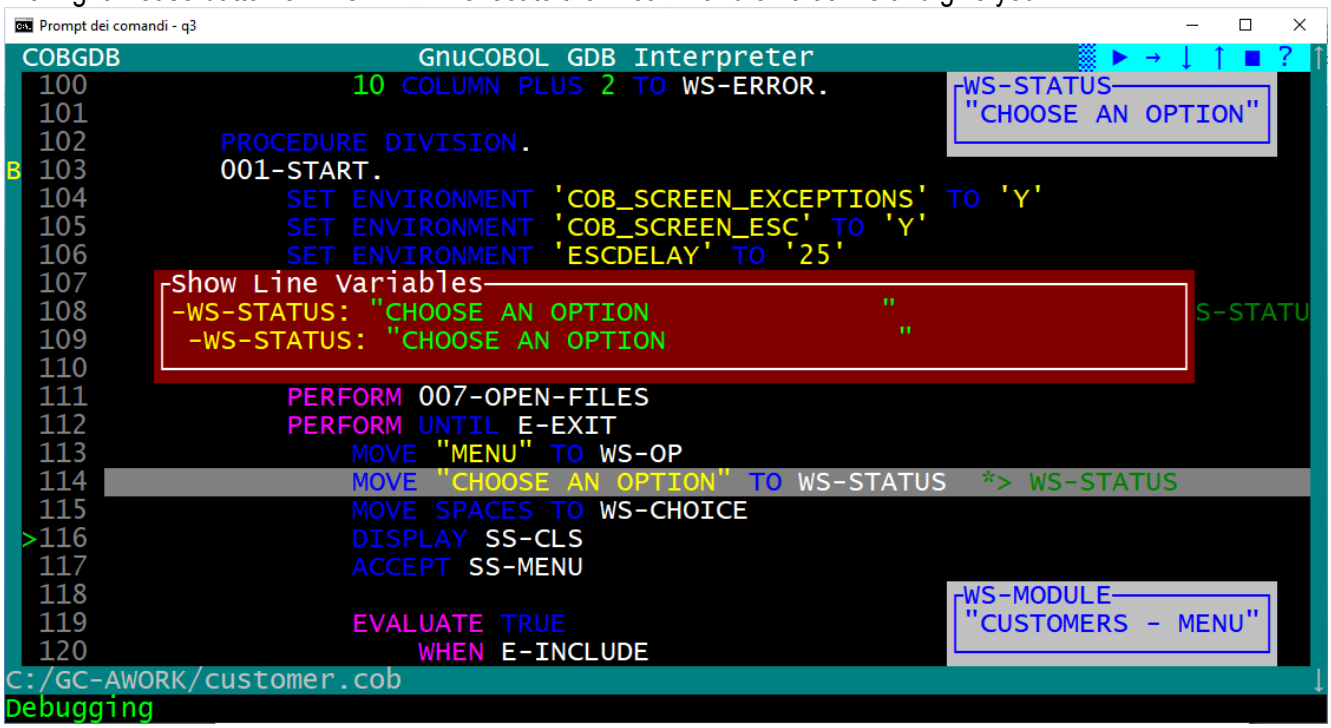


2.5. Show Command

typing the '**H**' command (key) allows you to view the variables on the highlighted line.

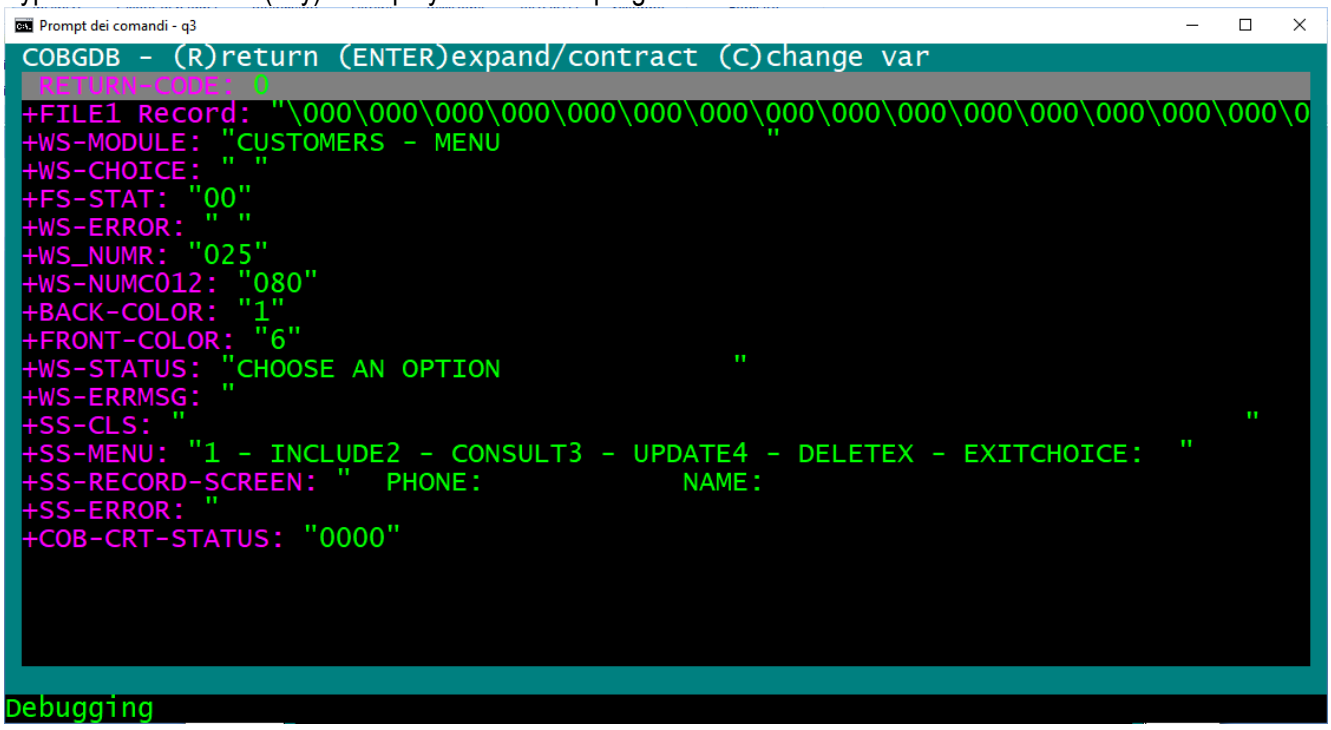


Note: to display the content of variables you can also click with right mouse button on a source line, example click with right mouse button on line 114 will execute the H command on that line and give you:



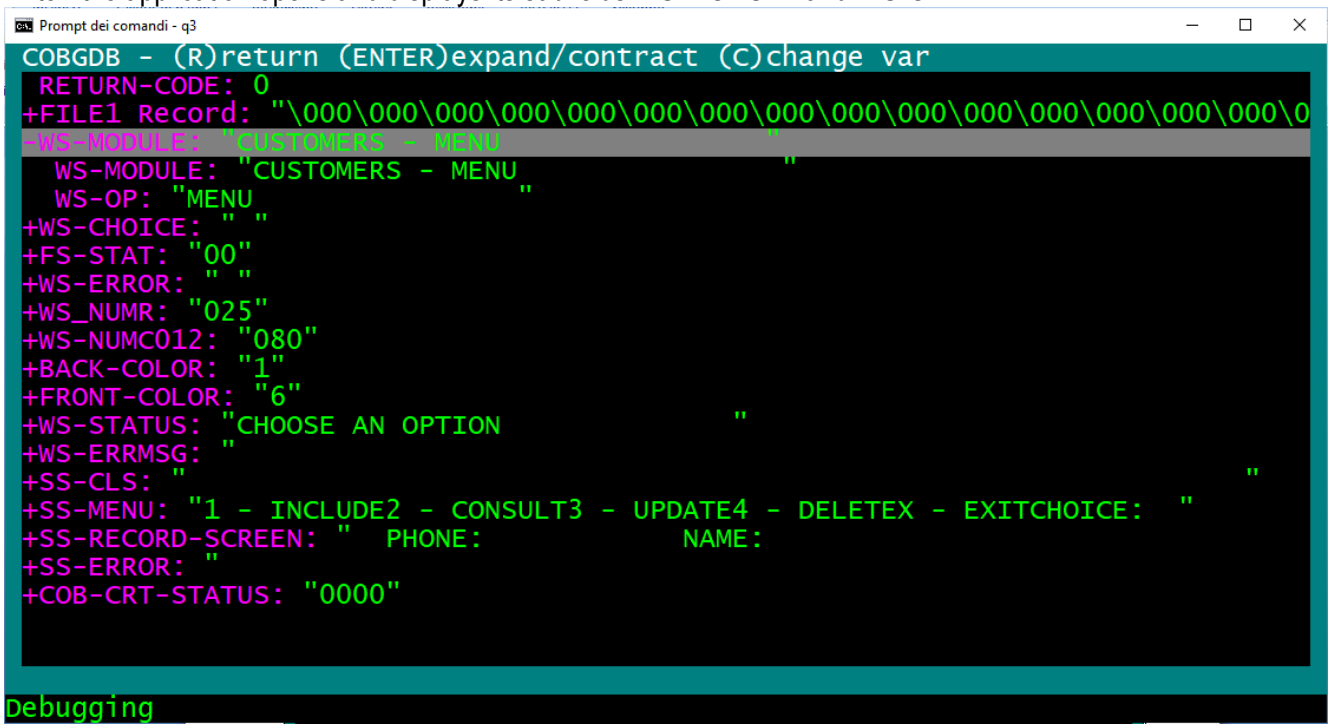
2.6. Variable Command

Type the **V** command (key) to display the list of all program variables:

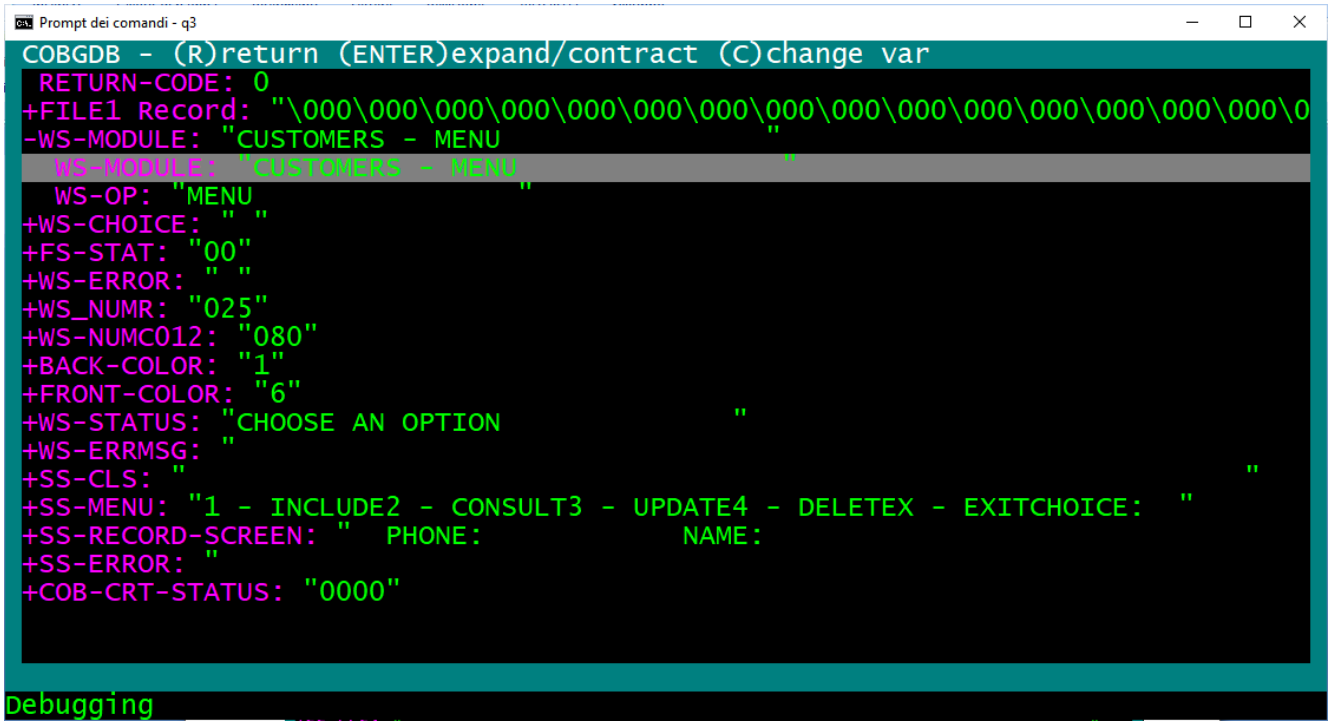


2.6.1. Enter subCommand

Scroll with cursor key UP and DOWN (or with mouse wheel) in this list to the variable WS-MODULE and type Enter: the application opens and displays its subfields WS-MODULE and WS-OP

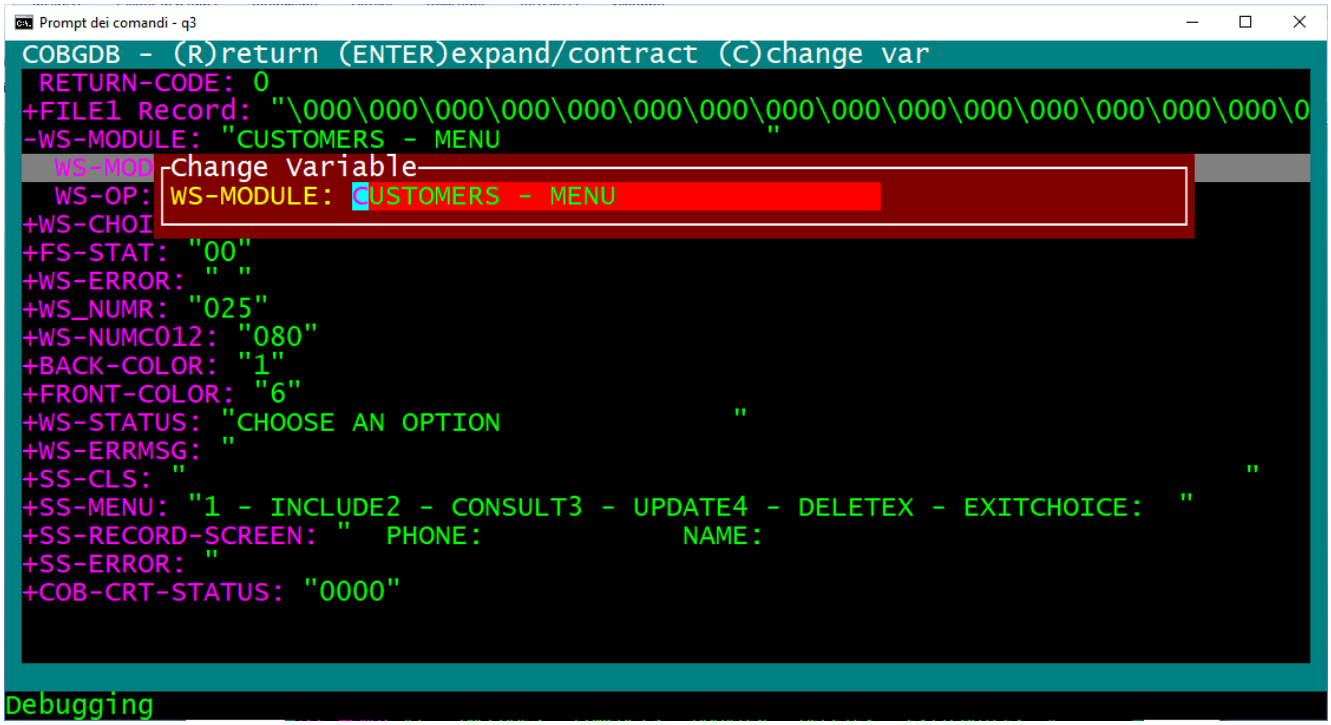


select WS-MODULE subfield

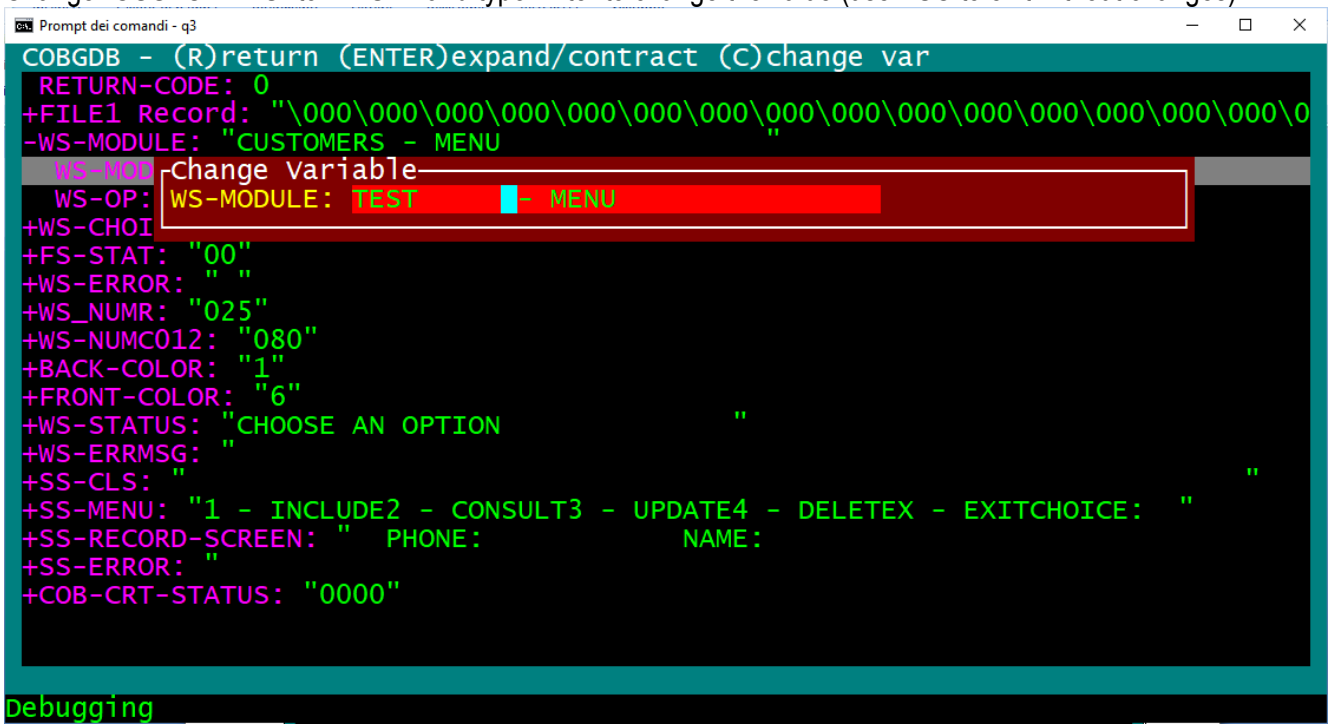


2.6.2. Change subCommand

Now you can select WS-MODULE subfield whit cursor DOWN or mouse wheel and type "**C**" (Change)
COBGDB shows a Change Variable window:



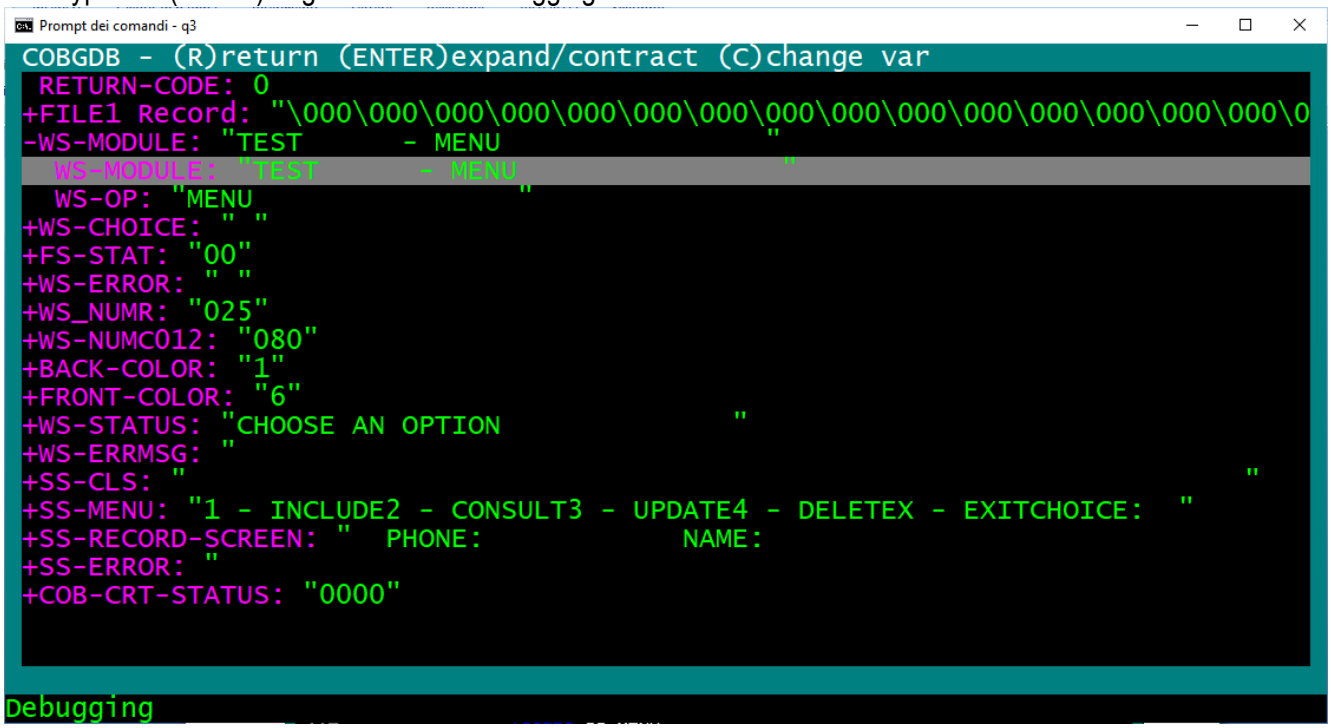
Change "CUSTOMERS" to "TEST" and type Enter to change the value (use ESC to exit without changes) :



2.6.3. Return subCommand

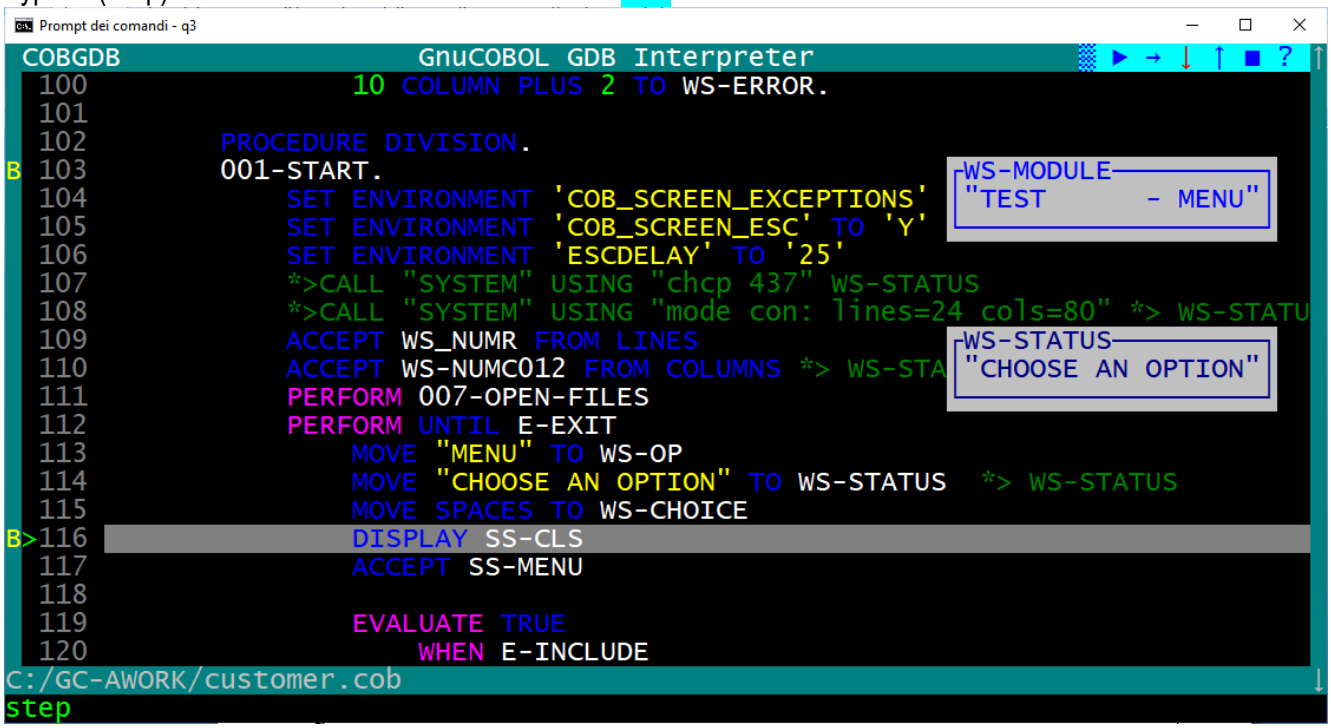
WS-Module has new value.

Now type "**R**" (Return) to go back to the debugging session:



2.7. Step Command

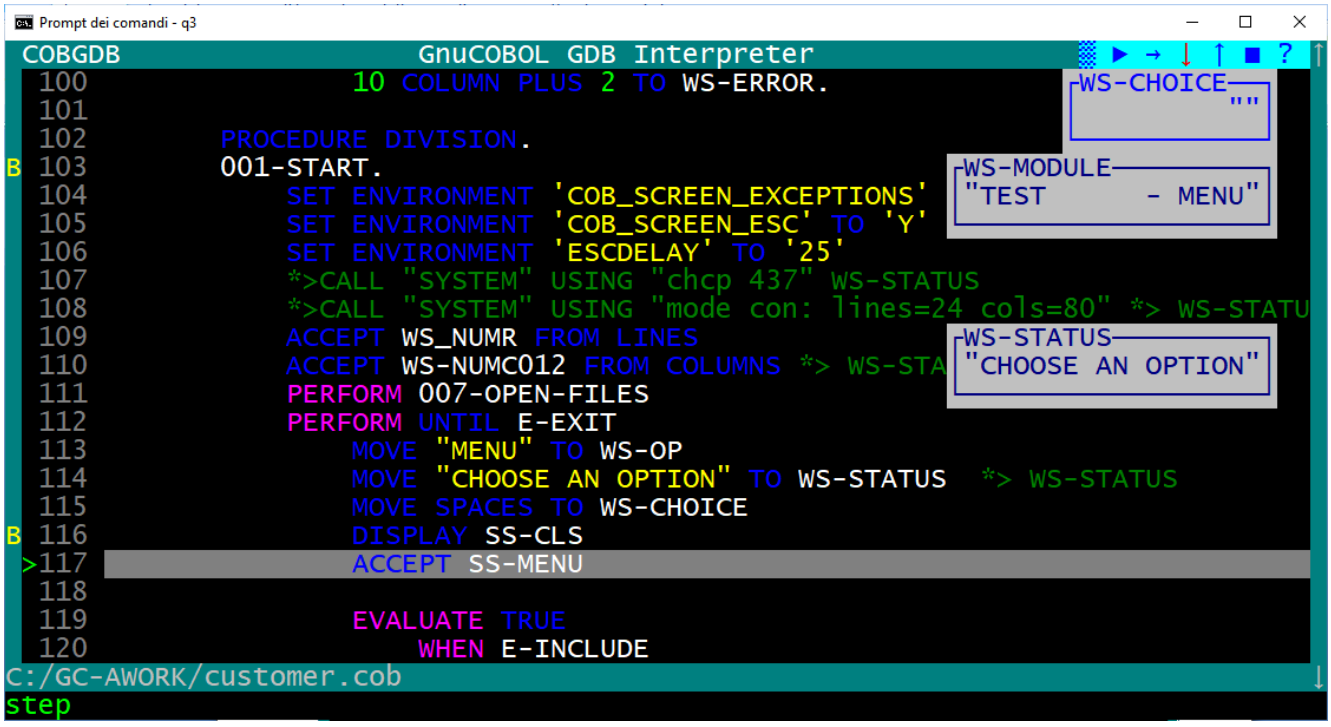
now you are back in the main debugging window:
Type **S** (Step) command or left click with mouse the ↓ button to execute the DISPLAY statement at line 116



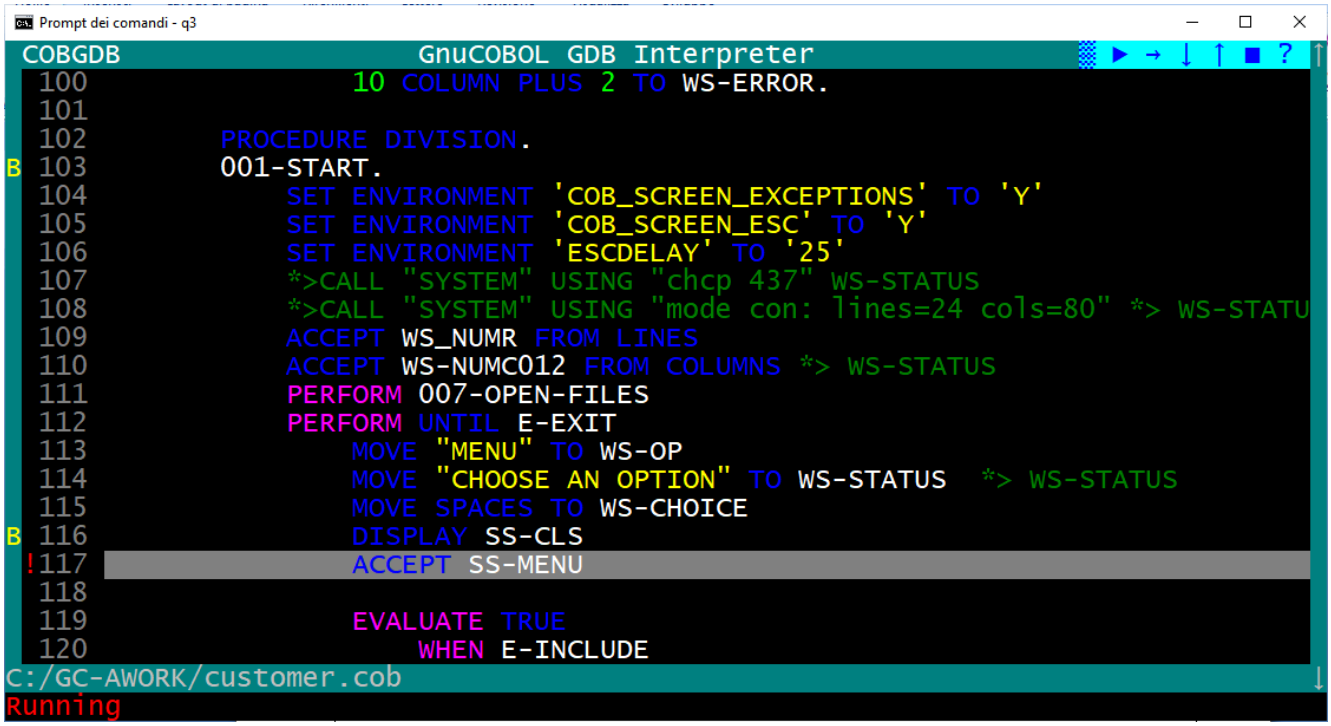
in the other application window you can see the result of DISPLAY statement

go back to debugging window and now the ACCEPT statement will be executed with S command

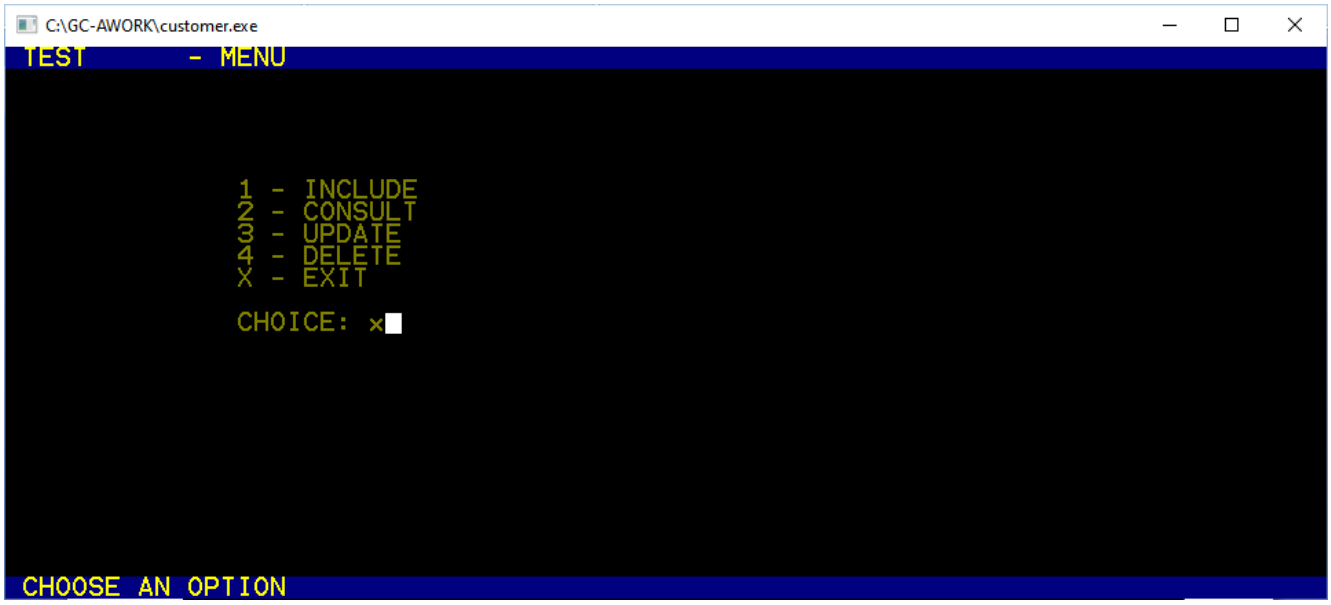
Type **S** (Step) command or left click with mouse the ↓ button again to execute the ACCEPT statement at line 117:



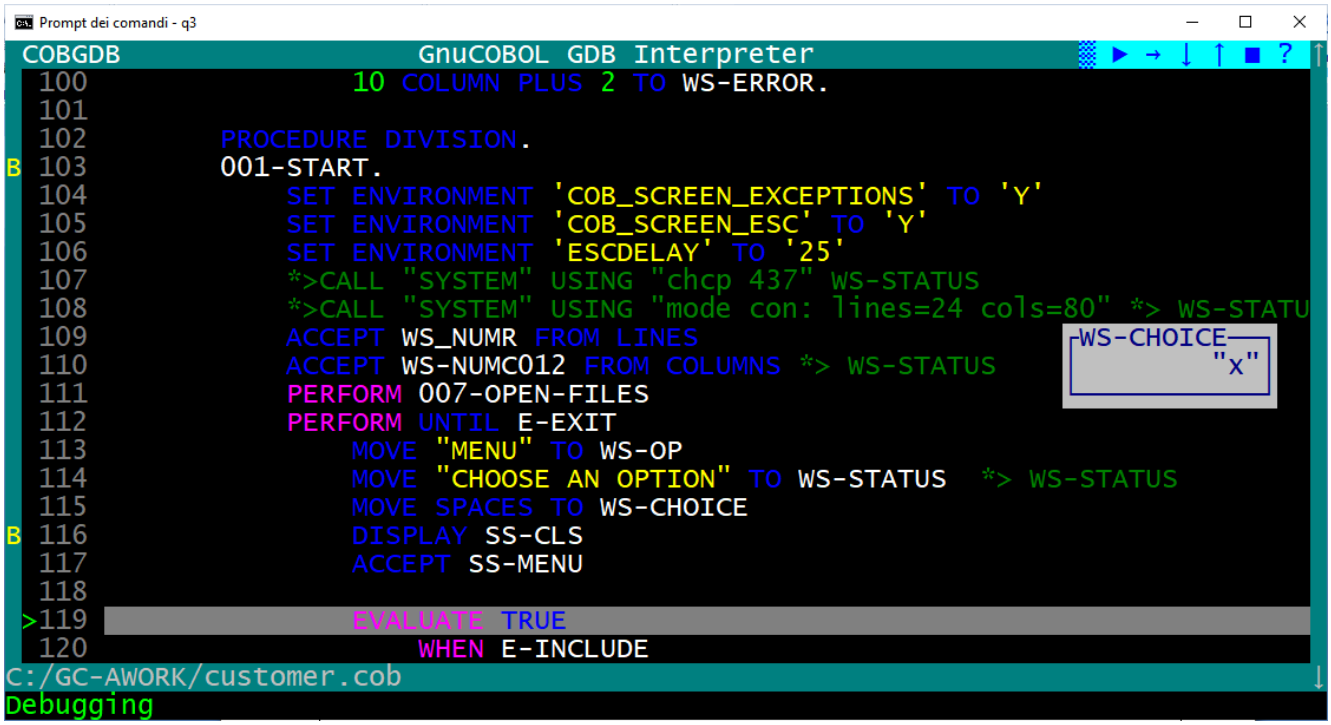
A red **!** quotation mark appears on the line ! 117 .
This means that application is running and a user action is required at application window.



The application is running in a separate window.
The 'Accept' command switches the focus to application windows.
After a user action on application screen (ex type the "X" choice and Enter) it is necessary to click again on the 'debugger' window to continue debugging.



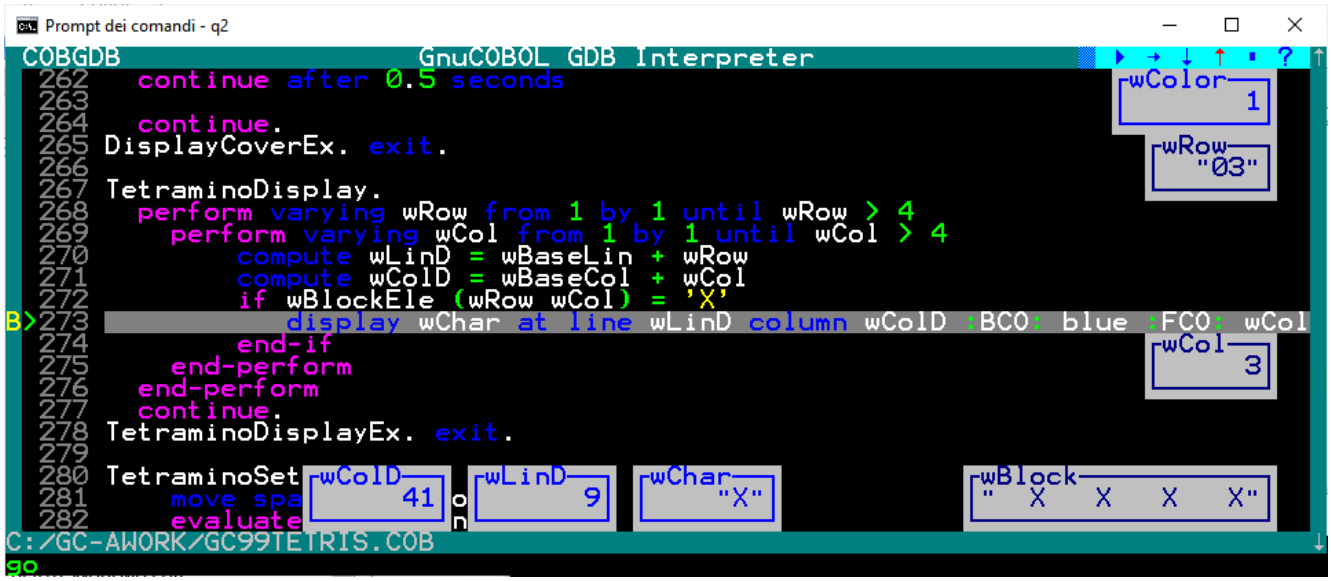
go back to debugger window: the ACCEPT statement has been executed:



2.8. Pop-up Variable windows

During a debugging session COBGDB shows variable content.
Blue frame and values : variables of executing cobol statement
Black frame and values : variables of last executed cobol statement.

Sample:



2.9. File Command

To show this command we use following sample:

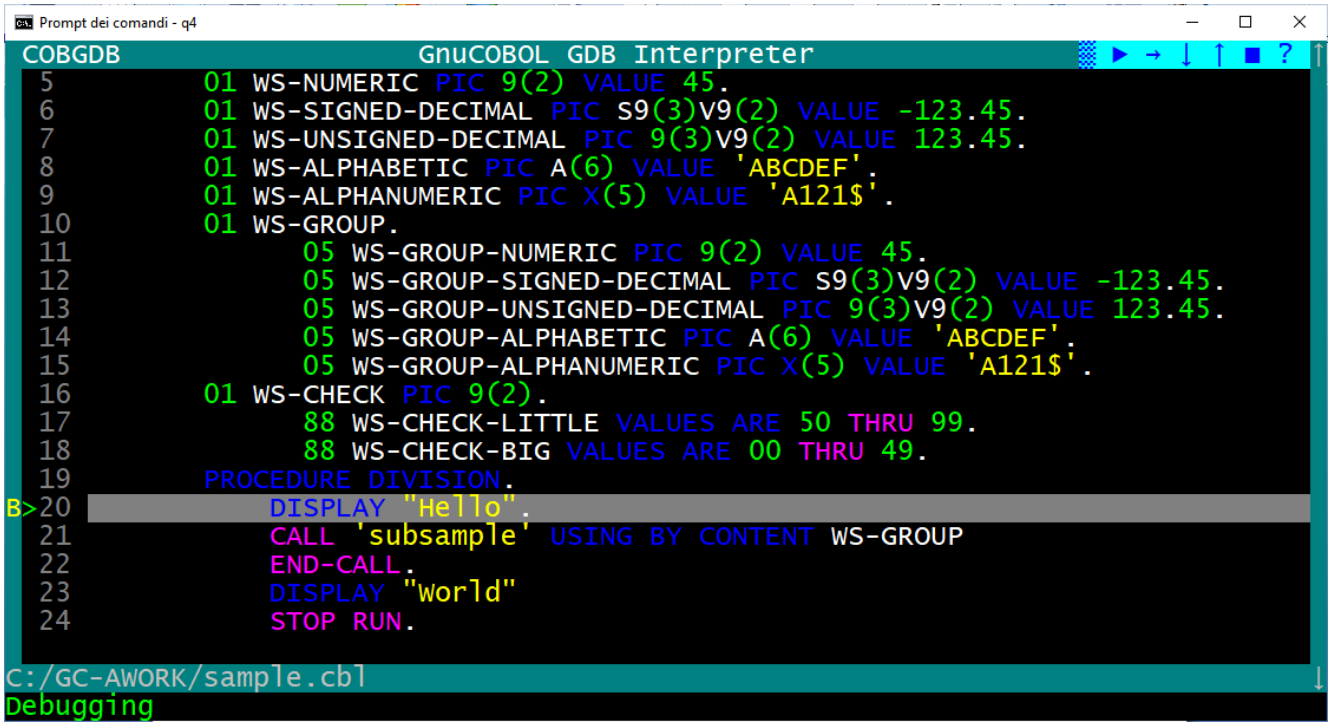
cobgdb sample.cbl subsample.cbl subsubsample.cbl -x -lpdcurses

where sample.cbl is the main program; it calls
--> subsample.cbl; it calls
--> subsubsample.cbl

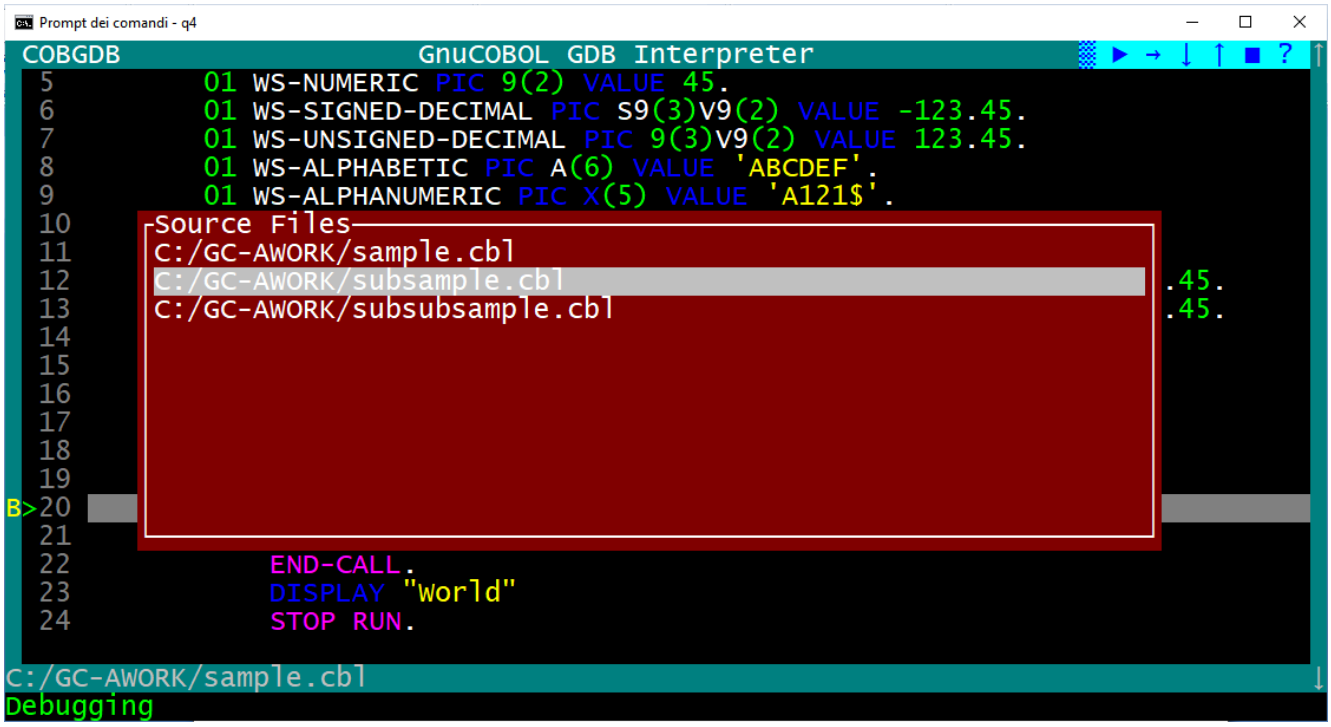
Source code is at https://github.com/marcsosduma/cobgdb/tree/main/resources.
This will create a single sample.exe executable.

This example shows that when you need to debug only subsample.cbl or only subsubsample.cbl you need to execute COBGDB with all three programs.

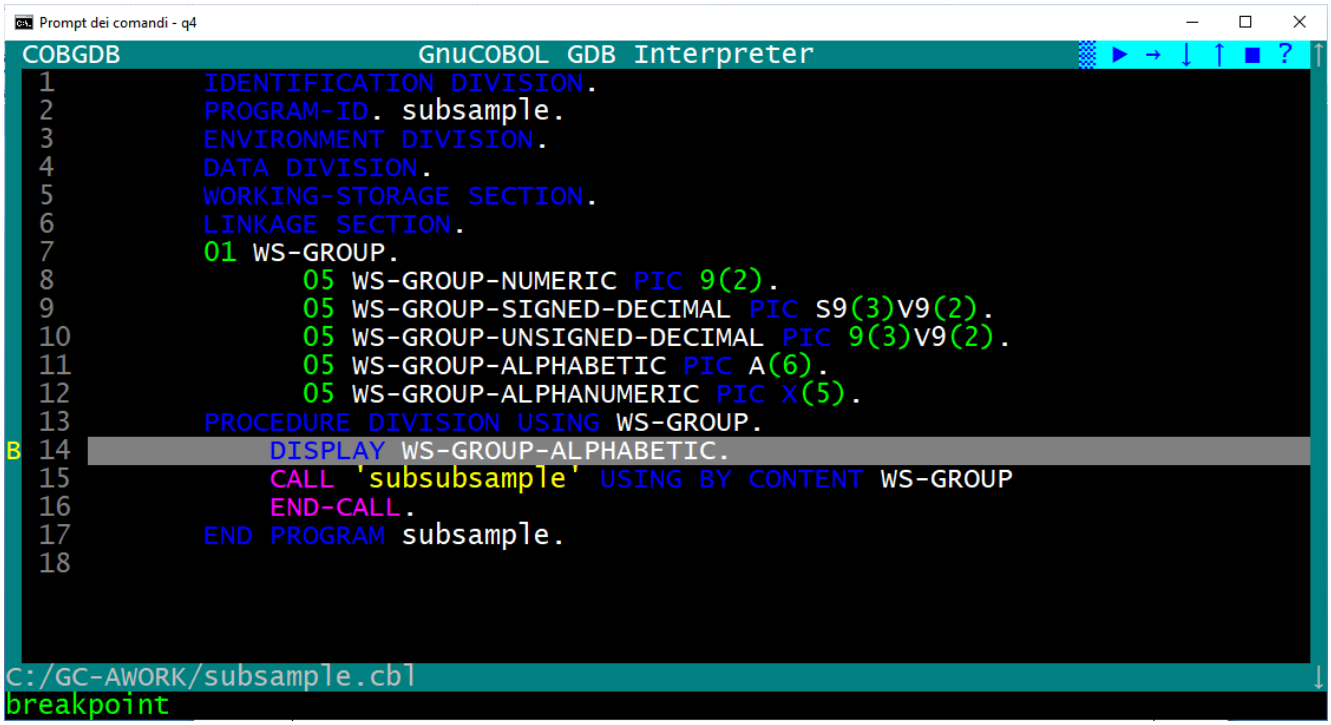
COBGDB sets the B breakpoint at first executable statement of first program "sample.cbl".
here use the R Run command to start the debugging session.



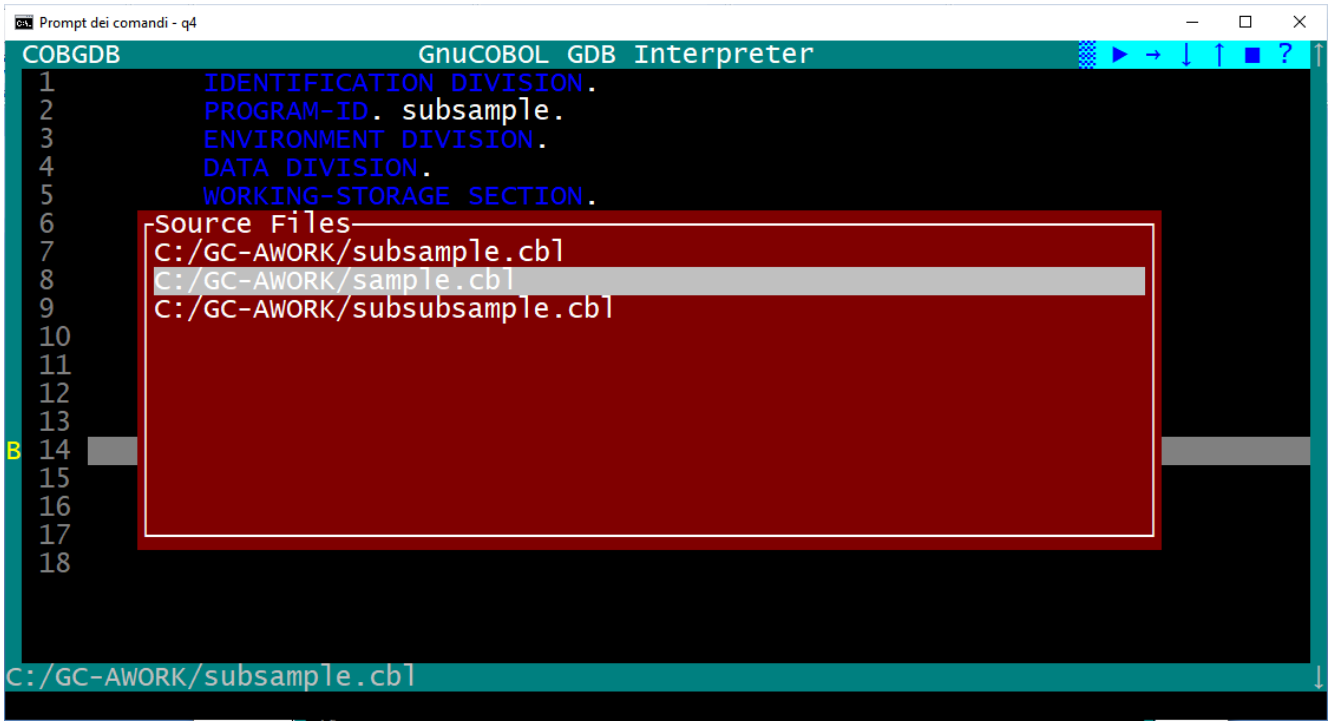
Now you can type the **F File** command and you will have the "Source Files" window.
In this sample we select the second program in the list (subsample.cbl) and type Enter.



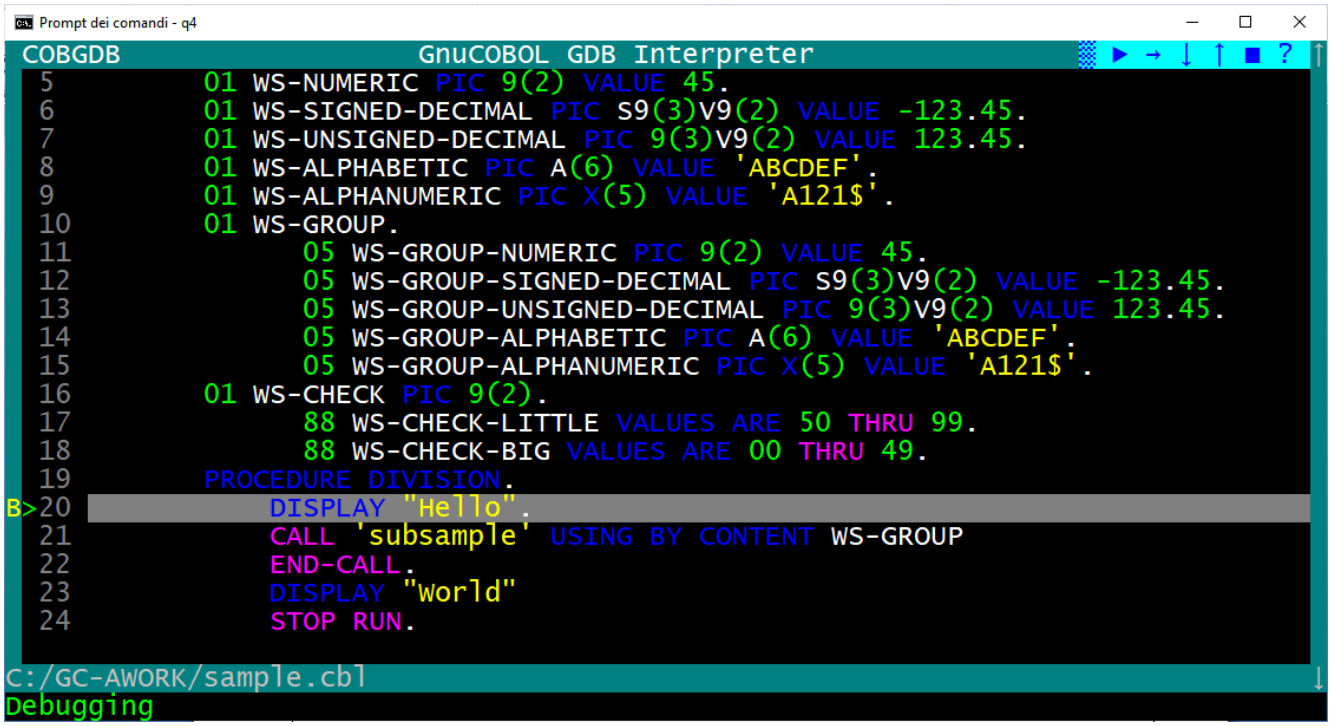
COBGDB shows the selected program source code where in this sample we type a B command at line 14.



now we type the F command again, then select the "sample.cbl" program and press Enter

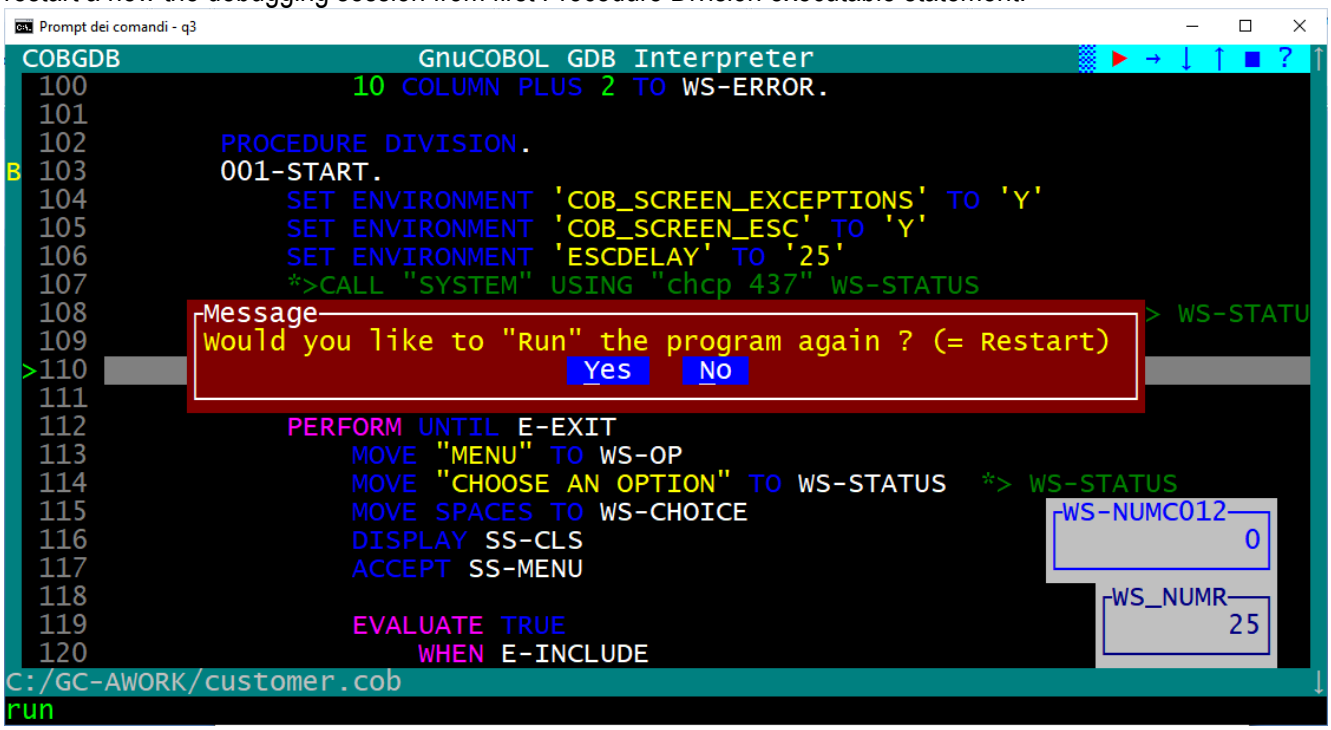


now we are back to the sample.cbl program to continue the debugging session as we need.

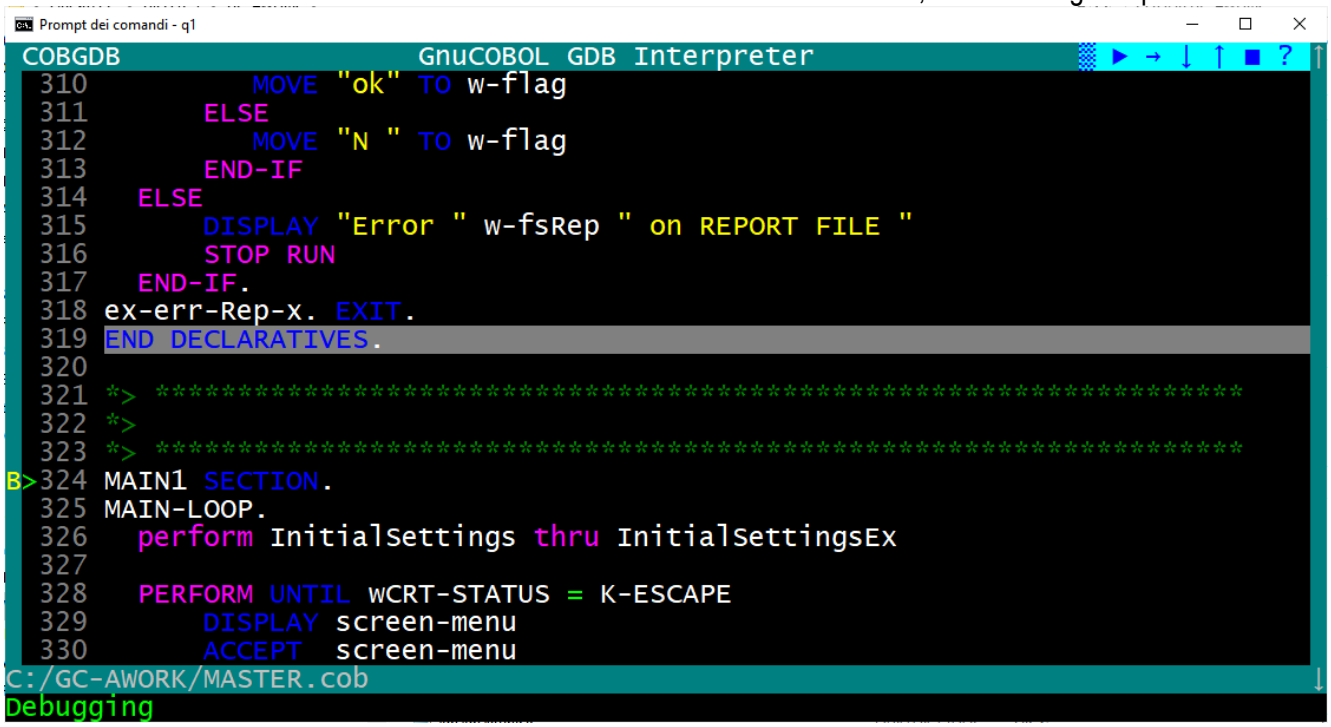


2.10. Run Command

If you click the Run command during a debug session you will receive a confirmation request, because Yes will restart a new the debugging session from first Procedure Division executable statement:

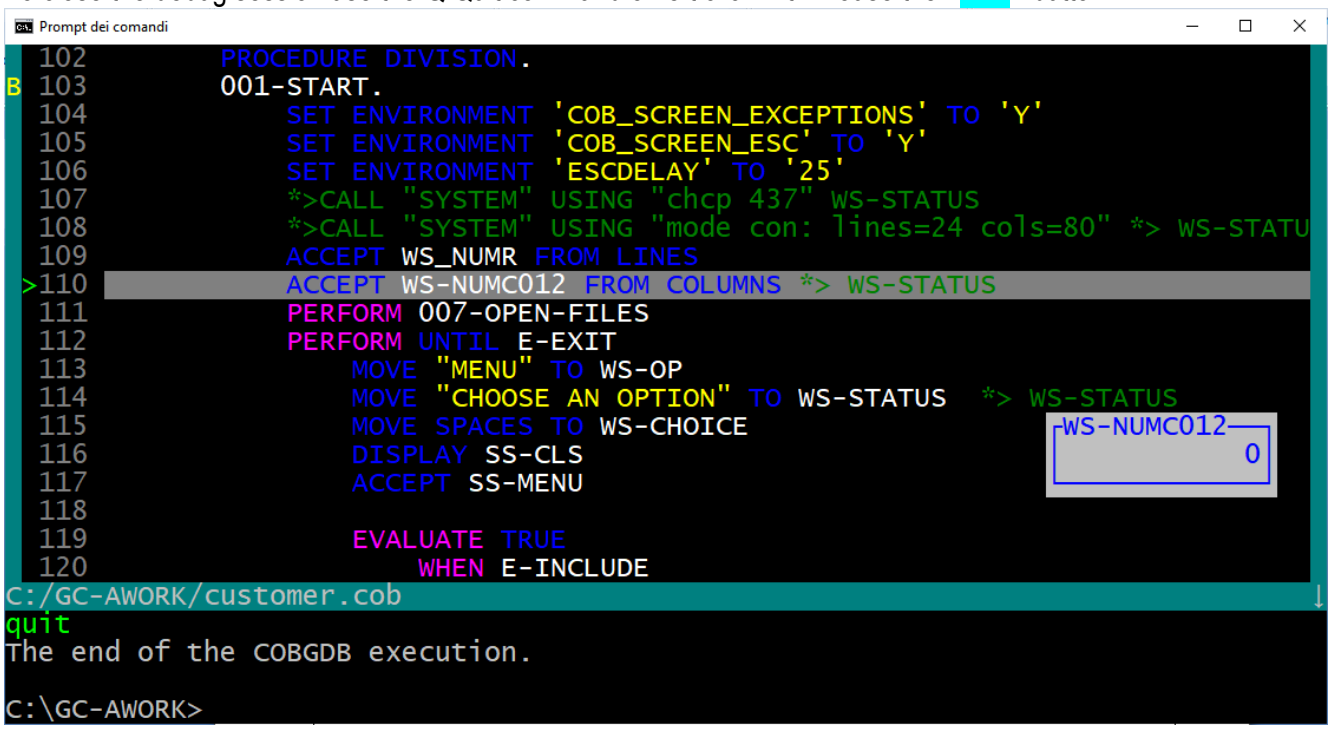


Note: if program has DECLARATIVES then the first automatic B Breakpoint will be settled at first executable PROCEDURE DIVISION statement that is the one after END DECLARATIVES, see following sample:



2.11. Quit Command

To close the debug session use the **Q** Quit command or left click with mouse the ■ button



3. Document Change Log

CHANGE LOG
Version1 of 2023.12.12. *First release* *. **Version2 of 2023.12.23.*** *Step by Step sample of use is added* *Some minor changes* *. **Version3 of 2024.0218.*** *Restructured showing new cobgdb screens and features*

Technical info